UNIVERSIDADE PAULISTA – UNIP EaD

Projeto Integrado Multidisciplinar

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

ALLAN CARVALHO BARBOSA RA: 0583355

BRUNO CESAR FIRMINO RA: 0584501

PROGRAMA DE CADASTRAMENTO DE PACIENTE EM ESTADO DE RISCO - SELFCARE

JACAREÍ
2020

ALLAN CARVALHO BARBOSA RA: 0583355

BRUNO CESAR RA: 0584501

PROGRAMA DE CADASTRAMENTO DE PACIENTE EM ESTADO DE RISCO - SELFCARE

Projeto Integrado Multidisciplinar para obtenção do título de tecnólogo Análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista – UNIP EaD.

Orientador(a):Orientador: Marcelo Henrique dos Santos.

JACAREÍ
2020

BARBOSA, Allan Carvalho; FIRMINO, Bruno Cesar. **Programa de cadastramento de paciente em estado de risco - selfcare**. 2020. 16 folhas. Projeto Integrado Multidisciplinar IV - Universidade Paulista, Jacareí, 2020.

RESUMO

O acompanhamento dos médicos para seus pacientes em relação ao Covid-19 criamos um aplicativo de fácil acesso aos pacientes e aos médicos para descobrir se os mesmos estão no estado de risco, o sistema foi desenvolvido pelo software SELFCARE, em conjunto com os programadores, alunos da UNIP, o aplicativo permite que sejam confirmados pelos médicos os números de pacientes que se encontra em risco com o objetivo do monitoramento é melhorar o atendimento dos médicos e também ter uma posição imediata do paciente, facilitando transferências e cuidados que podem ser necessárias por parte da regulação, o sistema vai gerar um cadastro completo de cada paciente utilizando algumas informações básicas e necessárias para cadastramento.

Palavras-chave: Software saúde, covid19, Selfcare, médicos.

BARBOSA, Allan Carvalho; FIRMINO, Bruno Cesar. **Program for registering patients at risk - selfcare.** 2020. 16 sheets. Multidisciplinary Integrated Project IV - Universidade Paulista, Jacareí, 2020.

ABSTRACT

The monitoring of doctors for their patients in relation to Covid-19 we created an application with easy access to patients and doctors to find out if they are at risk, the system was developed by the SELFCARE software, together with programmers, students from UNIP, the application allows doctors to confirm the numbers of patients who are at risk for the purpose of monitoring is to improve the care of doctors and also have an immediate position for the patient, facilitating transfers and care that may be needed by regulation, the system will generate a complete registration of each patient using some basic and necessary information for registration.

Key-words: Software health, covid19, Selfcare, doctors.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Em 2020 o mundo foi surpreendido por uma pandemia, causado pelo novo corona vírus; Doença essa que por sua vez está causando um grande impacto no sistema de saúde de vários países em todo o mundo.

Necessitando assim de novos métodos para se ter um controle de pessoas que são mais sugestivas ao vírus, que são em sua maioria idosos e pessoas com comorbidades.

Baseados nesse novo desafio enfrentado pelos profissionais de saúde, criamos um software para auxiliar esses profissionais de saúde a identificar esses pacientes.

O intuito do nosso software é auxiliar na compilação de dados do grupo de risco identificando onde tem uma maior concentração de pessoas de risco assim podendo ser tomada medidas previas para o combate e prevenção do vírus.

Nosso software leva em conta os dados relevantes do paciente, e baseado nessas informações nosso software identifica os pacientes de risco, e assim gerando um arquivo com dados do paciente e local onde reside, para ter dados estatístico e para um controle maior por parte do sistema de saúde.

No primeiro capitulo iremos abordar o método de desenvolvimento utilizado para a elaboração de nosso projeto, ficando a cargo do segundo capitulo o código fonte em si.

DESENVOLVIMENTO

Durante a fase de desenvolvimento optou-se por utilizar o Rapid Application Development (RAD), que é uma metodologia de grande sucesso em ambientes proprietários. Embora as ferramentas RAD livres ainda sejam desconhecidas por grande parte dos desenvolvedores, a sua utilização está ganhando força pela comunidade de software livre.

O foco para se utilizar no desenvolvimento RAD é a utilização de frameworks, onde esses estão disponíveis para desenvolvimento em linguagens como C e C++ sendo as mais utilizadas em ambientes baseados em software livre, embora estas linguagens não sejam tão produtivas para o desenvolvimento de aplicações rápidas.

O RAD foi registrado por James Martin, em 1991. É um processo de desenvolvimento de aplicações de forma rápida com objetivos bem definidos e análise de requisitos extremamente bem alinhada (GUEDES, 2020).

Esse modelo enfatiza um ciclo de desenvolvimento curto, com o intuito de ter um desenvolvimento melhor e mais rápido.

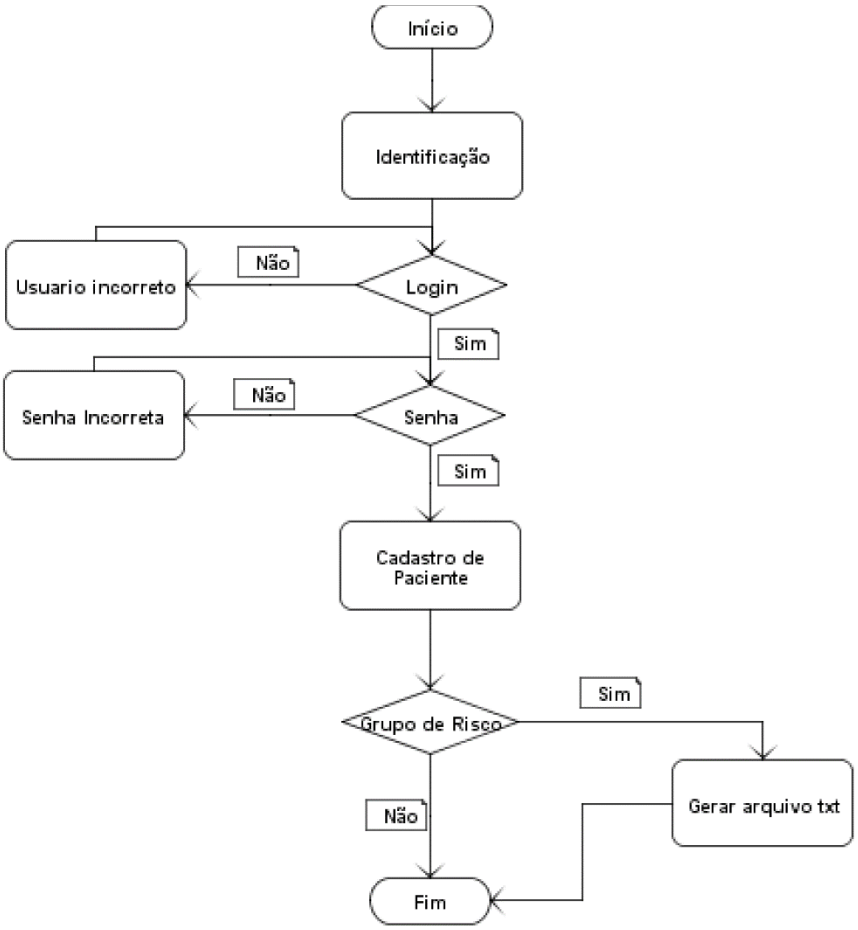
Segundo Guedes (2020), no desenvolvimento incremental, uma das características de RAD, o sistema é dividido em módulos, tomando por base a funcionalidade. Tendo os incrementos definidos, a cada ciclo é acrescido de novas funcionalidades ou até mesmo modificações, caso seja necessário. Outra característica é justamente essa maleabilidade de adaptação dos processos e a capacidade de se manter em constante evolução.

Um ponto muito importante é que a aplicação deve possuir requisitos muito bem definidos e o sistema poder ser modularizado para o bom funcionamento do RAD. Quando você analisar que o projeto a ser desenvolvido pode ser dividido em componentes e em ser reutilizado componentes prontos, sempre pensando no curto prazo, pode ser uma boa opção (GUEDES, 2020).

Baseado nesse método para o desenvolvimento do nosso projeto, pois o mesmo tem uma facilidade e prazo curto de entrega usamos a seguinte estratégia para o sistema de cadastro, enquanto um dos membros dos grupos estava desenvolvendo o sistema de login dos médicos e tomada de decisões do programa o outro estava fazendo o cadastramento dos pacientes, e o gerando os arquivos TXT para os pacientes de risco.

Para auxiliar em uma melhor visualização das funções e dos requerimentos necessários para o funcionamento do projeto, optou se pela criação de um fluxograma. Presente na figura 1 a seguir.

FIGURA 1 Fluxograma



Fonte: Autor deste trabalho (2020)

Assim podemos ter uma noção melhor de como o projeto seria feito, e também como poderia ser implementado com maior sucesso o método RAD e a divisão dos trabalhos pela equipe.

CÓDIGO FONTE

Para o desenvolvimento deste trabalho foi utilizado o compilador Code::blocks, ferramenta está muito completa para o desenvolvimento e execução do software, utilizando-se de linguagem C.

Optou-se por dividir o código em funções para o melhor desenvolvimento, pois cada função poderia ser codificada de maneira independente e por fim unificada.

Foi criado as seguintes funções, logar (), cadastro (), calculo_idade () e diagnostico ().

O programa inicia apresentando a função login (), responsável pela validação do usuário do sistema, que teve como primeiro desafio fazer a verificação dos dados digitados pelo usuário, com os dados predefinidos no código.

A solução encontrada foi utilizar a função strcmp (), que usa a biblioteca string.h, esta função compara duas strings e retorna um número inteiro, indicando a diferença numérica do primeiro caractere diferente da primeira cadeia em relação ao da segunda cadeia (MIZRAHI, 2008, p. 207). Na figura 2 observamos a aplicação da função.

FIGURA 2 – Função strcmp()

```
log = strncmp(login,aux,8);
```

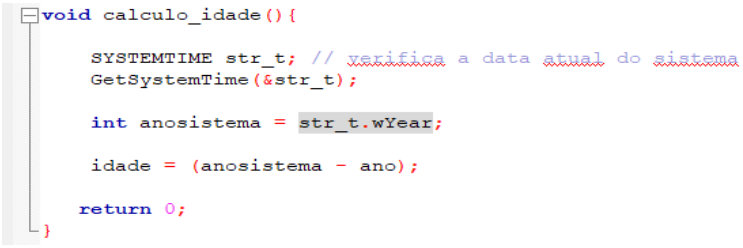
Fonte: Autor deste trabalho (2020)

A função cadastro () faz a parte de cadastro dos pacientes, recebe dados pessoais e clínicos do paciente e salva em variáveis, para que possa ser usado em outras partes do código.

Um dos requisitos do programa era o cálculo da idade do paciente, para a verificação se o mesmo estava no grupo de risco, ficando a cargo da função calculo_idade (), está por sua vez subtrai o ano atual pelo o de nascimento do paciente.

O desafio nesta parte foi em como buscar o ano atual no sistema, que foi resolvido com a introdução da biblioteca windows.h, utilizando-se da função SYSTEMTIME, especificamente para o ano str_t.wYear. Na figura 3 abaixo, observamos a o cálculo da idade.

FIGURA 3 – Função calculo_idade()



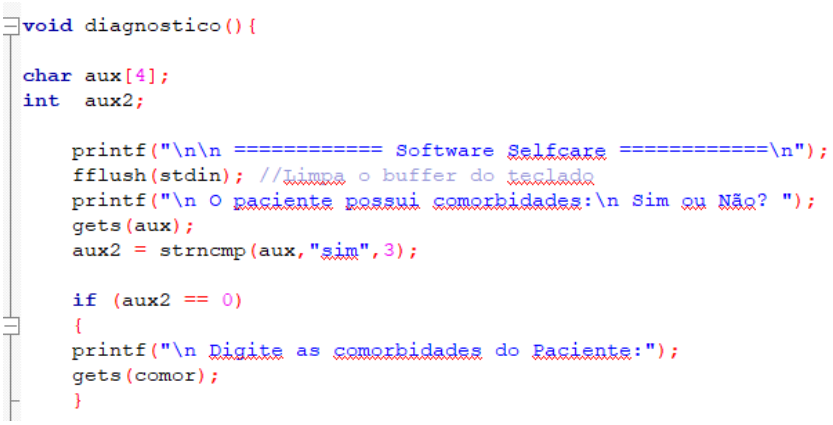
```
void calculo_idade(){

    SYSTEMTIME str_t; // verifica a data atual do sistema
    GetSystemTime(&str_t);

    int anosistema = str_t.wYear;

    idade = (anosistema - ano);

   return 0;
}
```

Fonte: Autor deste trabalho (2020)

Por fim, a verificação da comorbidade em conjunto com a idade, observando-se assim se o paciênte faz parte ou não do grupo de risco, está a cargo da função diagnostico ().

Nesta etapa o programa faz uma verificação se o paciente possui comorbidades, mais uma vez utilizando a função strcmp().Na próxima figura vemos a verificação se há comorbidades.

FIGURA 4 – Função diagnostico ()



```
void diagnostico(){

char aux[4];
int  aux2;

    printf("\n\n ============ Software Selfcare ============\n");
    fflush(stdin); //Limpa o buffer do teclado
    printf("\n O paciente possui comorbidades:\n Sim ou Não? ");
    gets(aux);
    aux2 = strncmp(aux,"sim",3);

    if (aux2 == 0)
    {
    printf("\n Digite as comorbidades do Paciente:");
    gets(comor);
    }
```

Fonte: Autor deste trabalho (2020)

Após isso através de uma comparação se há comorbidade e ou idade acima de 65 anos, os requisitos sendo satisfeitos o sistema gera um arquivo contendo o CEP e a idade do paciente.

Segundo MIZRAHI (2008) arquivos são usados para indicar um fluxo de bytes, ou seja, é um lugar na qual possa receber ou enviar bytes para a memória do computado, as informações necessárias são guardadas em uma estrutura do tipo

FILE. A figura 5 ilustra a parte responsável por gerar o arquivo txt contendo o CEP e idade do paciente.

FIGURA 5 – Código responsável por gerar um arquivo

```
    if (idade >= 65 || aux2 == 0) {
      FILE *selfcare; // Cria variável do tipo ponteiro para a criação do arquivo

      selfcare = fopen("Selfcare.txt", "a"); // Abrindo o arquivo

      fprintf(selfcare,"\n ============ Software Selfcare ============\n");
      fprintf(selfcare," CEP:%s ", cep);
      fprintf(selfcare,"\n Idade:%d ",idade);
      fclose(selfcare); //Fecha o arquivo
      printf("\n Arquivo gerado com sucesso\n");
      printf("\n ====== Obrigado por utilizar nosso programa ======\n");
    }
    else {
      printf("\n ====== Obrigado por utilizar nosso programa ======\n");
}
    return 0;
```

Fonte: Autor deste trabalho (2020)

Nas imagens a seguir iremos apresentar o Código fonte na integra, desenvolvido no compilador Code::Blocks.

FIGURA 6 – Código fonte software selfcare

```
/*
Software para inclusão de pacientes com Covid 19,
como requisito para o PIM IV, da Universidade Paulista - UNIP

Versão 1.0

Autores:

Allan Carvalho Barbosa   RA:583355
Bruno Cesar Firmino      RA:584501

Data da criação:     02/11/2020
Última atualização: 16/11/2020

*/
//================================================================================
//--- Inclusão das bibliotecas ---

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <windows.h>

//================================================================================
// --- Declaração das variáveis

int  senha    = 123456, aux, aux2, ano = 0, dia [3], mes [3], idade;
char login[8] = "usuario",comor[100], cep [9];

//================================================================================

int main()
{
    setlocale(LC_ALL,"portuguese"); // Define a linguagem de saída como Português

    printf("\n\n ============ Software Selfcare ============\n");
    printf("\n Seja bem vindo!\n");
    logar();
    system("cls");
    cadastro();
    system("cls");
    calculo_idade();
    diagnostico();

   system("pause");
```

```
47      return 0;
48  }
49  //=============================================================================
50  //--- Funções auxiliares ---
51
52  void logar()
53  {
54
55  char aux [8];
56  int log;
57  int aux2;
58
59      for(;;)
60   {
61
62       printf("\n Login:");
63       scanf("%s",&aux);
64       log = strncmp(login,aux,8);
65
66      if(log == 0)
67       {
68       printf("\n Senha:");
69       scanf("%d",&aux2);
70
71      if(aux2 == senha)
72          {
73
74       printf("\n Tudo certo!");
75       break;
76          }
77      else
78       printf("\n Senha incorreta, digite novamente.\n");
79       }
80      else
81       printf("\n Login incorreto, digite novamente.\n");
82   }
83  }
84
85  void cadastro(){
86
87  char nome[40], rua[40], bairro[40], cidade[40], uf[3], email[40];
88  int tel [14], num[6], ddata[12], cpf[12];
89
90      printf("\n\n ============ Software Selfcare ============\n");
```

```
91          printf("\n Digite os Dados do Paciente:\n");
92          fflush(stdin); //Limpa o buffer do teclado
93
94          printf("\n Nome:");
95          gets(nome);
96          fflush(stdin);
97
98          printf(" CPF:");
99          gets(cpf);
100         fflush(stdin);
101
102         printf(" Data de Nascimento (XX/XX/XXX):");
103         scanf ("%2d/%2d/%4d",&dia, &mes, &ano);
104         fflush(stdin);
105
106         printf(" Telefone:");
107         scanf ("%d",&tel);
108         fflush(stdin);
109
110         printf("\n\n Digite o Endereço:\n");
111         printf("\n Rua:");
112         gets(rua);
113         fflush(stdin);
114
115         printf(" Numero:");
116         gets (num);
117         fflush(stdin);
118
119         printf(" Bairro:");
120         gets(bairro);
121
122         printf(" Cidade:");
123         gets(cidade);
124
125         printf(" Estado [UF]:");
126         gets(uf);
127         fflush(stdin);
128
129         printf(" CEP:");
130         gets (cep);
131         fflush(stdin);
132
133         printf("\n Email:");
134         gets(email);
135         fflush(stdin);
```

```
136
137         printf("\n Data do diagnostico (XX/XX/XXX):");
138         gets (ddata);
139         fflush(stdin);
140         }
141     void calculo_idade(){
142
143         SYSTEMTIME str_t; // verifica a data atual do sistema
144         GetSystemTime(&str_t);
145
146         int anosistema = str_t.wYear;
147
148         idade = (anosistema - ano);
149
150        return 0;
151     }
152
153     void diagnostico(){
154
155     char aux[4];
156     int  aux2;
157
158         printf("\n\n ============ Software Selfcare ============\n");
159         fflush(stdin); //Limpa o buffer do teclado
160         printf("\n O paciente possui comorbidades:\n Sim ou Não? ");
161         gets(aux);
162         aux2 = strncmp(aux,"sim",3);
163
164         if (aux2 == 0)
165         {
166         printf("\n Digite as comorbidades do Paciente:");
167         gets(comor);
168         }
169
170         if (idade >= 65 || aux2 == 0) {
171           FILE *selfcare; // Cria variável do tipo ponteiro para a criação do arquivo
172
173           selfcare = fopen("Selfcare.txt", "a"); // Abrindo o arquivo
174
175           fprintf(selfcare,"\n ============ Software Selfcare ============\n");
176           fprintf(selfcare," CEP:%s ", cep);
177           fprintf(selfcare,"\n Idade:%d ",idade);
178           fclose(selfcare); //Fecha o arquivo
179
180              printf("\n Arquivo gerado com sucesso\n");
181              printf("\n ====== Obrigado por utilizar nosso programa ======\n");
182          }
183          else {
184            printf("\n ====== Obrigado por utilizar nosso programa ======\n");
185     }
186          return 0;
187     }//fim do programa
```

Fonte: Autor deste trabalho (2020)

Por fim, o programa apresentou um resultado satisfatório, cumprindo os pré-requisitos apresentados, demonstrou ser uma ferramenta simples, mas de grande eficiência na execução de sua função; foram necessárias 186 linhas de código.

CONCLUSÃO

Desenvolvimento do presente estudo possibilitou uma análise de como um software feito sob encomenda pode melhorar a apuração dos resultados dos pacientes de riscos. Além disso, também permitiu uma pesquisa de campo para obter dados mais consistentes sobre os pacientes.

Ao fazer a análise do nosso software percebemos que ela é de fácil aceso e que em questão de minutos já apontamos resultados significativos. Permitindo assim, que os objetivos propostos foram realmente alcançados.

Dada à importância do assunto e do momento que estamos passando hoje, torna-se necessário o desenvolvimento de formas de agilizar as partes mais demoradas da saúde e torná-las fácies de serem feitas digitalmente para pessoas sem muito conhecimento em informática. Podendo economizar não só o tempo como recursos naturais que são necessários para serem concluídas.

Nesse sentido, a utilização de recursos digitais permite aos operadores realizarem seu trabalho de forma mais rápida e eficiente; além disso, diminui o tempo de espera para pacientes realmente diagnosticados e dá a segurança necessária aos que não infectados, para que não se contamine, motivando as partes envolvidas.

REFERÊNCIAS

Blue Ink.biz. **Rapid Application Development**. Disponível em: < http://www.blueink.biz/RapidApplicationDevelopment.aspx >. Acesso em: 11 out. 2020.

MIZRAHI, Victorine Viviane. **Treinamento em linguagem C**. 2º Edição. São Paulo: Person Prentice Hall, 2008.

GUEDES, Marylene. **O que é RAD – Rapid Application Development.** Disponível em: < https://www.treinaweb.com.br/blog/o-que-e-rad-rapid-application-development/ >. Acesso em: 05 nov. 2020.